HUMAN MOTION FOLLOWING COMPUTER MOUSE AND GAME CONTROLLER

Related Applications

This application is a continuing application of commonly-owned and co-pending U.S. provisional application number 60/070,512, filed on January 6, 1998, and U.S. provisional application number 60/100,046, filed on September 11, 1998, each of which is incorporated herein by reference.

Background

The primary human interfaces to today's computer are the keyboard, to enter textual information, and the mouse, to provide control over graphical information. These interfaces help users with word processing, presentation software, computer aided design packages, spreadsheet analyses, and other applications. These interfaces are also widely used for computer gaming entertainment; though they are often augmented or replaced by a joystick.

In daily use of business software applications, control of cursor position on the screen requires that the user remove his/her hand from the keyboard in order to use the standard mechanical mouse. The use of the mouse introduces several issues. In a desk environment, the mouse requires maintenance of space on the desk area. The mouse cord must also remain free from obstruction to facilitate movement. Additionally, the use of the mouse is a major contributing factor of carpal-tunnel

syndrome. It would be advantageous therefore to find an alternative to the mechanical mouse.

In computer gaming, game complexity generally requires control of the (i) mouse and keyboard, or (ii) joystick and keyboard. Further, gaming applications usually require control in several axes of motion, including forward motion, reverse motion, left turn, right turn, left strafe (slide), right strafe, upward motion, downward motion. To further complicate game maneuvers and control, many games permit viewing (within the game environment) in directions different from that in which the vehicle (e.g., the car, or person, simulated within the game) is moving, including up, down, left and right. These many complexities of motion in-fact increase or modify the complexity and enjoyment of the game.

Nevertheless, these complexities require that the user have utmost dexterity and control of his/her body. One object of the invention, therefore, is to offer alternative approaches to human-computer interfaces for those incapable of using standard devices (e.g., mouse, keyboard and joystick) such as due to disability.

Another object of the invention is to provide an alternative input device for laptop computers. Laptop computers are used in locations which do not allow the use of a mouse, in airplanes or during business meetings in which there is no room to operate the mouse. Through the use of either a clip on camera or a camera built into the laptop display, the laptop user can control the mouse position or use the camera for teleconferencing while on the road.

Other objects of the invention are to replace or augment existing human computer interfaces to facilitate enhanced gaming and/or control within game environments.

In the prior art, certain systems exist which attempt to reduce the amount of physical interaction required with game controllers. However, such systems are

prohibitively expensive to the general public as their costs are driven by techniques and algorithms which detect user head motion based upon a detectable target worn by the user. Other costly and cumbersome systems require the user to wear apparatus which emits or detects a signal. It is thus one other object of this invention to provide a system which detects user motion without the aid or augmentation of artificial devices placed on the user operator.

Another object of the invention is to provide a means of human control of a graphical computer interface through the physical motion of the user in order to control the activity of a cursor in the manner usually accomplished with a computer mouse.

A further object of the invention is to provide additional degrees of freedom in the human computer interface in support of computer games and entertainment software.

Yet another object of the invention is to provide dual use of teleconferencing and video electronics with gaming and computer control systems.

These and other objects will be apparent in the description which follows.

Summary of Invention

As used herein, "cursor" means a computer cursor associated with a computer screen. "Scene view" means the view presented on a computer display to a user. For example, one scene view corresponds to the scene presented to a user during a computer game at any given moment in time. The game might include displaying a scene whereby the user appears to be walking in a forest, and through trees. In another example, a cursor might also be visible in the scene view as a mechanism for the user to select certain events or items on the scene (e.g., to open a door in a game, or to open a folder to access computer files).

As used herein, "camera" refers to a solid state instrument used in imaging. Typically, the camera also includes optical elements which refract light to form an image on the camera's detector elements (typically CCD or CMOS). For example, one camera of the invention derives from a video-conferencing camera used in conjunction with Internet communication.

In one aspect, the invention provides systems and methods to control computer cursor position (or, for example, the scene view or game position as displayed on the computer display) by motion of the user at the computer. A camera rests on or near to the computer, or built into the computer, and connects therewith to collect "frames" of data corresponding to images of the user. These images provide information about user motion, over time. Software within the computer assesses these frames and algorithmically adjusts cursor motion (or scene view, or mouse button, or some other operation of the computer) based upon this motion. The motion may be imparted by up-down or left-right motion of the user's head, by the user's hands, or by other motions presented to the video camera (such as discussed herein). In one aspect, a close up view of the users facial features is used to impart a translation in the cursor (or scene view) even through the features in fact rotate with the user's head. In yet another aspect, the rotation is used to generate a corresponding rotation in computer game scene imagery.

In one aspect, the invention also provides a human factors approach to cursor movement in which the user's rate of motion determines the relative motion of the cursor (or scene view). By way of example, the faster the user's head travels over a set distance, the further the corresponding cursor movement over the same time period.

In other aspects of the invention, the camera is either (a) a visible light camera utilizing ambient lighting conditions or (b) a camera sensitive in another band such as the near infrared ("IR"), the IR, or the ultraviolet ("UV") spectrum. In the latter

case (b), the illumination preferably emanates from a source such as an IR lamp which is beyond human sensory perception. The sensor is typically mounted facing the user so as to capture a picture of the user's face in the associated electromagnetic spectrum. The lamp is typically integrated with the camera housing so as to facilitate production and ease of consumer set-up.

In one aspect, a system of the invention provides an IR camera (i.e., a camera which images infrared radiation) to image the user's face and to gauge the user's stress level associated with a game on the computer. As the user's intensity increases (such as in a fast moving computer game using a joystick or the methods discussed herein), the system detects increased heat intensity on the user's face, forehead or other body part by the imagery of the IR camera. This information is fed back into the game processor to provide further enhancement to the game. In this manner, the system gauges the user's reaction to the game and modifies game speed or operation in a meaningful way. For example, suppose such a system determined that a particular user was bored of the present game speed (a determination of boring can be made by assessing low IR output over large portions of the user's face). The computer processor and game software can then cooperate to increase the gaming speed and thereby increase this particular user's stress. Games of the invention are thus made and sold to users with varying intelligence, age and/or computer familiarity; and yet the system always "pushes the envelope" for any given user so as to make the game as interesting as possible, automatically.

In accord with one aspect of the invention, images captured by the sensor are processed by a digital signal processor ("DSP") located either (a) in a PC card within the host computer or (b) in a housing integrated with the sensor. In case (a), sensor frames are sent to the PC card; and detected user motion (sometimes denoted herein as "difference information") is communicated to the user's operating system via a PCI (or USB or later standard) bus interface. These difference information commands are interpreted by a low overhead program resident at the user's main processor, which either updates the cursor position on the screen or provides motion

information to the user's computer game (e.g., so as to change the scene view). In case (b), the DSP is contained within the camera housing; and frames are processed local to the camera to determine difference information. This information is then transmitted to the computer by a cable that connects to a bus port of the computer so that the host processor can make appropriate movements of the cursor or scene view. In another aspect, the DSP is mounted in the camera housing such that the camera/signal processing subsystem produces signals which emulate the mouse via the mouse input connector.

In an alternative configuration, frames of image data are sent directly to the host computer through the computer bus; and that image data is manipulated by the computer processor directly. With increasing computer processing speed, it is expected that sensor data frames can be sent directly to the host processor for all processing needs, in which case the PC card and/or separate DSP are not required. Although this is possible today, the update rates are likely too slow for practicality. Once GHz processors are on the market, a separate DSP may no longer be needed.

In one aspect of the invention, pixel format or pixel density of the camera drives the accuracy of the system. Higher pixel density in the image of the user's face, for example, increases the attainable resolution and cursor control (or the attainable control of scene view motion). Camera formats of 240 vertical by 320 horizontal generally provide satisfactory performance. The number of pixels that may be utilized is determined by system cost factors. Greater numbers of pixels require more powerful DSPs (and thus more costly DSPs) in order to process the image sequences in real time. Current technology limits the processing density to a 64x64 window for consumer electronics. As prices are reduced, and power increases, the densities can increase to 128x128, 256x256 and so on. While 64x64 density is satisfactory for general household users, a higher fidelity system using a greater number of pixels is possible, in accord with the invention, for higher end applications at a proportionally higher cost. Non-square pixel formats are also possible in accord with the invention, including a 64x128 detector array size.

In one aspect, the data transfer rate from the camera is 30 frames/second at 240x320 pixels per frame. Assuming eight bits per pixel, the digital data transfer rate is therefore 18.432 megabits/second. This is a fairly high transfer rate for consumer products using current technology. While the data transfer can be either analog or digital, the preferred method of image data transfer for this aspect is via a standard RS170 analog video interface.

In accord with one aspect, a system of the invention defines two imaging zones (either within a single camera CCD or within multiple CCD cameras housed within a single housing). One imaging zone covers the user's head; and the other covers the user's eyes. This aspect includes processing means to process both zones whereby movement of the user's head provides one mechanism to control cursor movement (or scene view motion), and whereby the user's eyes provide another mechanism to control the movement. In essence, this aspect increases the degrees of freedom in the control decision making of the system. By way of example, a user might look left or right within a game without moving his head; but by assessing movement of the user's eyes (or the pupils of those eyes), the scene view can be made to rotate or translate in the manner desired by the user. Further, a user might move his head for other reasons, and yet not move her eyes from a generally forward looking position; and this aspect can assess both movements (head and eyes) to select the most appropriate movement of the cursor or scene view, if any.

In another aspect, a system of the invention utilizes a camera with zoom optics to define the user's pupil and to make cursor or scene views move according to the pupil. In another aspect, the system incorporates a neural net to "learn" about a user's eye movements so that more accurate movements are made, over time, in response to the user's eye movement.

In still another aspect, a neural net is used to learn about other movements of the user to better specify cursor or scene view movement over time.

In yet another aspect of the invention, a system is provided with two CCD arrays (either within a single camera body or within two cameras). The arrays connect with the user's computer by the techniques discussed herein. One CCD array is used to image the user's head; and the other is used to image the user's body. Motion of the user is then evaluated for both head and body movement; and cursor or scene view movement is adjusted based upon both inputs.

In another aspect of the invention, a single CCD is used to image the user. However, alternate frames are zoomed, electronically, so that one frame views the user's head, and the next frame views the user's eyes. With the algorithm discussed herein, these separate frame sequences (one for the eyes, one for the head) are processed separately and evaluated, together, to make the most appropriate cursor or scene view movement. If for example, the system clocks at 30Hz, then one set of frame sequences operates at 15Hz, and the other at 15Hz. However, the advantage is that two movement information sets can be evaluated to invoke an appropriate movement in the cursor or scene view.

Those skilled in the art should appreciate that different frame rates can be used; and frame rates for either sequence (head or eyes) can occur at different rates too. Further, the separate frame sequences can utilize other body parts, e.g., the head and the hand, to have two movement evaluations. Alternatively, a separate camera (or CCD array) can be used to image other body parts, for example one camera for the head and one for the hand.

The invention also provides methods for shifting cursor or scene views in response to user movement. In one aspect, the scene view shifts left or right when the user shifts left and right. In another aspect, the scene view rotates when the user's head rotates. This last aspect can be modified so that such rotation occurs so long as the eyes do not also rotate (in this situation, the user's head rotates, indicating that she wishes the scene view to rotate; but the eyes do not, indicating

that she still watches the game in play). In another aspect, the scene view rotates in response to the user's hand rotation (i.e., a camera or at least a CCD array of the system is arranged to view the player's hand).

In another aspect, the invention provides a multi-zone player gaming system whereby the user of a particular computer game can select which zone operates to move the cursor or the scene view. By way of example, the system can include one zone corresponding to a view of the user's head, where frames of data are captured by the system by a camera. Another zone optionally corresponds to the user's hand. Another zone optionally corresponds to the user's eyes. Each zone is covered by a camera, or by a CCD array coupled within the same housing, or by optical zoom zones within a single CCD, or by separate optical elements that image different portions of the CCD array. By way of example, two zones can be covered with a single CCD array (i.e., a camera) when the zones are the user's head and eyes. The camera images the head, for one zone, and images the eyes in another zone, since the zones are optically aligned (or nearly so). However, two cameras (or optionally two CCD arrays with separate optics) can view two zones such as the user's head and the user's hand. Combinations of zones is also possible and envisioned in accord with the invention.

Zones in a single camera can also be identified by the computer by prompting the user for motion from corresponding body parts. For instance, the computer identifies the head zone by prompting the user to move his head. Then the computer identifies the foot zone by having the user move his foot. Once the zones are identified, the motion of each of these individual zones is tracked by the computer and the regions of interest in the camera image related to the zones moved as the targets in the zones move with respect to the camera.

In one aspect, the invention provides a system, including a camera and edge detection processing subsystem, which isolates edges of the user's body, for example, the side of the head. These edges are used to move the cursor or scene

view. For example, if the left edge of head is imaged onto column X of one frame of the CCD within the camera, and yet the edge falls in column Y in the next frame, then a corresponding movement of the cursor or scene view is commanded by the system. For example, movement of the edge from one column to the next might correspond to ten screen pixels, or other magnification. In one aspect, this magnification is selected by the user. Up and down motion can also be detected by similar edge detection. For example, by imaging the user's chin, an edge movement in the up or down dimension is formed (e.g., if the bottom edge of the chin moves from one row to the next, in adjacent frames, then a corresponding movement of the cursor or scene view is made - magnification again preferably set manually with a default starting magnification). Other images can also serve to define edges. For example, in one aspect, a user's eyelash can be used to move the cursor (or scene view) up or downwards; though typically the eye blink is used to reset the cursor command cycle.

In one aspect, an optical matched filter is used to center image zones onto the appropriate images. For example, as discussed above, one aspect preferably utilizes 64x64 pixels as the image frame from which cursor motion is determined. Many cameras have, however, many more pixels. These 64x64 arrays are therefore preferably established through matched filtering. By way of example, an image of a standard pair of user's eyes is stored within memory (according to one aspect of the invention). This image field is cross-correlated with frames of data from the actual image from the camera to "center" the image at the desired point. With eyes, specifically, ideally the 64x64 sample array is centered so as to view both eyes within the 64x64 array. Similarly, to process sequences of head data, a standard head image is stored within memory, according to one aspect, and correlated with the actual image to center the head view.

Those skilled in the art should appreciate that an appropriate frame size can be established from an image having more or fewer pixels, by redundantly allocating data into adjacent pixels or by eliminating intermediate pixels, or similar technique.

In another aspect, a camera is provided which optically "zooms" to provide optimal imaging for a desired image zone. By way of example, the invention of one aspect takes an image of the user's head, determines the location of the user's eyes (such as by matched filtering), and optically zooms the image through movement of optics to provide an image of the eyes in the desired processing size format.

Many aspects of the invention are preferably enhanced by autofocus. Specifically, it is often desirable to have a crisp image of the user (or a part of the user, e.g., the user's eyes) in order to accurately process desired cursor or scene view movement. Thus, autofocus capability preferably operates in most of the aspects of the invention where imaging is a feature of the processing.

In one aspect, the camera utilizes a very small aperture which results in a very large depth of field. In such a situation, autofocus is not required or desired. The optical requirements for the lenses are also reduced.

The invention thus provides several advantages over the art. For example, game controllers can now include feedback corresponding to the user's actual movement. By way of another example, if the user moves left or right (or head or hand or eyes move left or right, depending on the image zone), then the cursor (or scene view) can also be set to move left or right. When the user twists her head, for example, the scene view can also be made to rotate, reflecting that movement.

Those skilled in the art should appreciate that the direction in which the scene moves, left or right, is a matter of design choice. That is, certain games might find it desirable to move the opposite direction from what the user moves, to add certain challenges to the game. Further, in other aspects, this direction can change during the game to further complicate game control.

In accord with one aspect of the invention, a processing subsystem (connected with the camera) is used to make cursor movement (or scene view movement)

correspond to user's motion. This processing subsystem of another aspect further detects when the user twists his head, to add an additional dimension to the movement.

In one aspect, a system of the invention includes an IR detector which is used to determine when a person sweats or heats up (by imaging, for example, part of the user's head onto the IR detector); and then the system adjusts game speed in a way corresponding to this movement. Alternatively, a heartbeat sensor is tied to the person to sense increased excitement during a game and the system speeds or slows the game in a similar manner. Note that a heartbeat sensor can be constructed, in one aspect of the invention, by thermal imaging of the user's face, detecting blood flow oscillations indicative of heartbeat. In other aspects, the heartbeat sensor is physically tied to the user, such as within the computer mouse or joystick.

In one aspect, a computer of the invention adapts to user control as selected by a particular user. For example, in the case of a handicapped person, a particular user might select certain hand-movements, e.g., a single finger up, to move the cursor up; and another finger down to move the cursor left. An infinite combination of controls can be established; however this is one advantage of the invention in that users with many different disabilities can program cursor or scene view movement. In one aspect, a neural network is used to assist the processing system in establishing proper cursor movement. In another aspect, the computer for example learns to print something by movement of the user's finger (or other body part).

In one aspect, tipping of the user's head (or other body part, or object) is used to provide another degree of freedom in moving the cursor or adjusting the scene view. By way of example, a tilt of the head, as imaged by the camera, can be set to command a rotation of the scene view.

In still another aspect, a camera of the invention uses autozoom to move in and out of a given scene view. By way of example, the camera is first focussed on the

user's face in one frame; but in a subsequent frame the camera must focus to closer to compensate for the fact that the user moved closer to the camera (typically, the camera is on the monitor, so this also means that the user moved closer to the scene view). This autofzoom is used, in one aspect, to make the scene view appear as if the user is "creeping" into the scene. By moving the scene in and out, the user will perceive that he is moving in or out of the scene view.

In another aspect, a camera images an object held by the user. Preferably, the object has a well-defined shape. The system images the object and determines difference information corresponding to movement of the object. By way of example, rotating the object upside down results in difference information that is upside down; and then the scene view inverts by operation of the system. In another example, twisting of the object rotates the scene view left or right, or rotates the scene in the direction of the twisting.

In another aspect, two cameras image the user: one camera pointed at the front of the users face or hand and the other down at the top of the users head or hand. The front facing camera is used to detect rotational and linear translation in up-down and left-right directions. The top viewing camera determined front-back, left right translation. The front–back translation observed by the top camera is used to control forward and back motion in the users 3-D view. The top sensed left-right translation controls the users left right slide or strafe. The top sensed left-right motion is removed from the front view left-right translation with the remaining front view measure representative of left-right twist. All of the front view up-down translation can be interpreted as up-down twist.

Brief Description of the Drawings

Figure 1 illustrates one human computer interface system constructed according to the invention;

Figure 1A shows an exemplary computer display illustrating cursor movement made through the system of Figure 1;

Figure 1B illustrates overlayed scene views, displayed in two moments of time on the display in Figure 1, of a shifting scene made in response to user movement captured by the camera of Figure 1;

Figure 1C shows an illustrative frame of data taken by the system of Figure 1;

Figure 2 illustrates selected functions for a printed circuit card used in the system of Figure 1;

Figure 3 illustrates an algorithm block diagram that preferably operates with the system of Figure 1;

Figure 4 illustrates one preferred algorithm process used in accord with the invention to determine and quantify body motion;

Figure 5 shows one process of the invention for communicating body motion data to a host processor, in accord with the invention, for augmented control of cursor position or scene view;

Figure 5A shows representative frame of data of a user taken by a camera of the invention, and further illustrates adding symbols to key body parts to facilitate processing;

Figure 6 illustrates a two camera imaging system for implementing the teachings of the invention;

Figure 7 illustrates two positions of a user as captured by a camera of the invention; and Figure 7A illustrates two positions of a scene view on a display as repositioned in response to movement of the user illustrated in Figure 7;

Figure 8 illustrates motion of a user – and specifically twisting of the user's head - as captured by a system of the invention; Figure 8A illustrates a first scene view corresponding to a representative computer display before the twisting; Figure 8B illustrates a second scene view corresponding to a rotation of the first scene view in response to the twisting by the user; Figure 8C shows processing features of the processing section of Figure 8; and Figure 8D illustrates multiple image frames stored in memory for matched filtering with raw images acquired by the system of Figure 8;

Figure 9 illustrates a two camera system of the invention for collecting N zones of user movement and for repositioning the cursor or scene view as a function of the N movements; Figure 9A illustrates a representative thermal image captured by the system of Figure 9; and Figure 9B illustrates process methodology for processing thermal images as a real time input to game processing speed, in accord with the invention;

Figure 10 illustrates another two camera system of the invention for targeting multiple image movement zones on a user, and further illustrating optional DSP processing at the camera section;

Figure 11 illustrates framing multiple movement zones with a single imaging array, in accord with the invention;

Figure 12 illustrates framing a user's eyes in accord with the invention; and Figure 12A shows a representative image frame of a user's eyes;

Figure 13 illustrates one system of the invention, including zoom, neural nets, and autofocus to facilitate image capture;

Figures 14, 14A and 14B illustrate autofocus motion control in accord with the invention;

Figures 15 and 15A illustrate one other motion detect system algorithm utilizing edge detection, in accord with the invention;

Figure 16 illustrates one other motion detect system algorithm utilizing well-characterized object maniputions , in accord with the invention;

Figure 17 illustrates one other motion detect system algorithm utilizing varied body motions, in accord with the invention;

Figure 18 illustrates a two camera system of the invention with a camera observing the user's face while the other observes the top of the user's head;

Figure 19 shows a blink detect system of the invention; and

Figure 20 shows a re-calibration system constructed according to the invention.

Detailed Description of the Drawings

Figure 1 illustrates, in a top view, certain major components of a human computer interface system 10 of the invention. A user 12 of the system 10 sits facing a computer monitor 14 with display 14a. A camera 16 is mounted on the computer monitor 14 facing the user 12. In the illustrated embodiment, the camera 16 is mounted in such a way that the user's face 12a is imaged within the camera's field of view 16a. However, as discussed herein, the camera 16 can alternatively image other

locations, such as the user's hand, eyes, or on other objects; so imaging of the user's face, in Figure 1, must be considered illustrative, rather than limiting. Further, the camera location can also reside at places other than on top of the monitor 14.

With further regard to Figure 1, the camera 16 interfaces with a printed circuit card 18 mounted within the user's computer chassis 20 (which connects with the monitor 14 by common cabling 20a). The camera 16 interfaces to the printed circuit card 18 via a camera interface cable 22. The circuit card 18 also has processing section 18a, such as a digital signal processing ("DSP") chip and software, to process images from the camera 16.

In operation, the camera 16 and card 18 capture frames of image data corresponding to user movement 25. The processing section 18a algorithmically processes the image data to quantify that movement 25; and then communicates this information to the host processor 30 within the computer 20. The host processor 30 then commands movement of the computer cursor in a corresponding movement 25a, Figure 1A (Figure 1A illustrates a representative front view of the display 14a, and also illustrates movement 25a of the cursor 26 moving within the display 14a in response to user movement 25).

Figure 1B illustrates an alternative (or supplemental) process whereby the scene view shifts in response to user movement 25. Specifically, Figure 1B illustrates a first scene view 35a, which generally corresponds to a forest prior to the user's movement 25; and an overlayed scene view 35b (shown in dotted line, for purposes of illustration) that is shifted by an amount 37 in response to the user's movement 25. The shift 37 in the scene view 35 is accomplished by combined operation and processing of the processing section 18a and host CPU 30.

Figure 1C shows a representative frame 41 of data 43 as taken by the camera 16. As illustrated, data 43 represents the user's face 12a taken at a given moment of time. Subsequent frames (not shown) are used to determine user motion 25 relative

to the frame 41, as discussed herein. The frame 41 is made up of the plurality of pixel data 45, as known in the art.

Figure 2 illustrates certain functions processed within the printed circuit board 18 of Figure 1. A camera interface circuit 50 receives video data from the camera 16 through interface cable 22. This video data can be RS170 format or digital, for example. For analog RS170 format, circuit 50 decodes the analog video data to determine video timing signals embedded in the analog data. These timing signals are used for control of the analog-to-digital (A/D) converter included in circuit 50 that converts analog pixel data into digital images. In the preferred embodiment, the analog data is digitized into 6-bits, though any number of bits greater may be acceptable and/or required for features as discussed herein. For digital data format, camera interface 50 accepts the digital data without additional quantization, although interface 50 can digitally pre-process the digital images if desired to acquire desired image features.

The frame difference electronics 52 receives digital data from the camera interface circuit 50. The frame difference electronics 52 include a multiple frame memory, a subtraction circuit and a state machine controller/memory addresser to control data flow. The frame memory holds previously digitized frame data. As each digitized pixel is received by the frame difference electronics 52, the corresponding pixel from a previous frame is read from the frame memory and subtracted from the current frame. The preferred implementation uses the frame just previous to the current frame, though an older frame which resides in the frame memory may be used. The resulting difference is output to an N-frame video memory 54. The new frame pixel data is then stored into the frame memory of the frame difference electronics 52.

The N frame video memory electronics 54 either receives differenced frames output by the frame difference electronics 52 (discussed above) or raw digitized frames from the camera interface 50. The choice of where the data derives from is

made by software resident on the DSP 56. The frame video memory 54 is sized to hold more than one full frame of video and up to N number of frames. The number of frames N is to be driven by hardware and software design.

In the preferred embodiment, the DSP 56 implements an algorithm discussed below. This algorithm determines the rate of head motion of the user in two dimensions. The digital signal processor 56 also detects the eye blink of the user in order to emulate the click and double click action of a standard mouse button. In support of these functions, the DSP 56 commands the N frame video memory 54 to supply either the differenced frames or the raw digitized frames. The digital signal processor thus preferably utilizes a supporting program memory 58 made up of electrically reprogrammable memory (EPROM) and data memory 59 including standard volatile random access memory (RAM). The DSP 56 also interfaces to the PCI bus interface electronics 60 through which cursor and button emulation is passed to the user's main processor (e.g., the CPU 30, Figure 1). The PCI interface 60 also passes raw digitized video to the main processor as an optional feature. Interface 60 also permits reprogramming of program memory 58, to allow for future software upgrades permitting additional features and performance.

The PCI interface electronics 60 thus provides an industry standard bus interface supporting the aforementioned communication path between the printed circuit card 18 and the user's main processor 30.

With optional MPEG compression electronics 62, the printed circuit card 18 and camera 16 can provide compressed video to the user's main processor 30. This compressed video supports using the system 10 in teleconferencing applications, providing dual use as either human computer interface system 10 and/or as a teleconferencing system in an economical solution to two distinct applications.

Figure 3 describes one preferred head motion block diagram algorithm 70 used in accord with the invention. Not all of the functions shown in Figure 3 are

implemented in software in the DSP 56. Rather, this algorithm relies on the correlation of images from one frame to the next, and particularly relies on the use of frame differenced images in the correlation process. The frame differencing operation removes parts of the camera images that are unchanged from the previous frame. For example, room background (such as object 13, Figure 1) behind the user 12 is removed from the image. This greatly simplifies detection of feature motion. Even the image of the user's face image consists of regions of uniform illumination such that even with the user's facial motion, these uniform regions (i.e. cheeks, forehead, chin) may also be removed. The user's face 12a also consists of typically dynamic features such as the nose, eyes, eyebrows and mouth, each of which typically has enough spatial detail that will be evident in the differenced image. As the user moves his face with respect to room lighting, the shape and distribution of these features will change; but the frame rate of the camera 16 ensures that these features look similar from one frame to the next. The correlation process therefore operates to determine how these differenced features are moving from one frame to the next in order to determine user head motion 25.

The algorithm of block diagram 70, Figure 3, receives video images 72 of the user as imaged by camera 16 over time. Each received image is passed to both a frame memory 74 and a differencer 76. Though the preferred embodiment is to buffer a single frame in memory 74, the memory 74 may optionally store many frames, buffered such that the first frame input is the first frame output. The delayed frame is read from the frame memory 74 and subtracted from the current frame using the differencer 76. Frame output from the differencer 76 is provided to both a correlation process 78 and a difference frame memory 80.

Like the frame memory 74, the preferred embodiment of frame memory 80 utilizes a single difference frame; however the difference frame memory 80 can hold many difference frames in sequence for a finite time period. The delayed difference frame is read from the difference frame memory and provided to the correlation function 78. Difference frames are preferably selectable by system algorithms. The

correlation process 78 determines the best combination of row and column shifts in order to minimize the difference between the current difference frame and the delayed difference frame. The number of rows and columns required to align these difference images provides information as to the user's motion.

The best-fit function algorithm 82 determines the row and column shift to provide optimal alignment. In the case of a classical correlation process, the best-fit function can consist of a peak detect algorithm. This algorithm may either be implemented in hardware or in software.

The best-fit function algorithm determines relative motion in rows and columns of the observed user's features. The cursor update compute function algorithm 84 translates this measured motion into the position change required of the cursor (e.g., the cursor 26, Figure 1A). Typically, this is a non-linear process that, with greater head motion, the cursor moves a non-proportionally greater distance. For example a 1-pixel user motion can cause the cursor to move one screen pixel while a 10-pixel user motion may cause a 100-pixel screen cursor motion. However, these magnifications can be adjusted for desired result. This algorithm may either be implemented in hardware or in software such as through an ASIC or FPGA.

Video cursor control 86 provides a user interface to enable and disable the operation of cursor control described above. This control is implemented, for example, through a combination of keystrokes on the user's keyboard (for example as connected to the host computer 20, Figure 1). Alternatively, cursor control is activated or deactivated by sensing the eye-blink of the user (or some other predetermined movement). In this alternative embodiment, an output signal 85 from the correlation section 78 is sent to the video enable section 86; and the output signal 85 corresponds to blink data from the user's face 12a (Figure 1A). In another embodiment, the video cursor control section 86 activates or deactivates cursor control by recognizing voice commands. A microphone 87 detects the user's voice and a voice recognition section 89 converts the voice to certain activate or deactivate

signals. For example, the section 89 can be set to respond to "activate" as a voice command that will enable cursor control; and "deactivate" as a command that disables cursor control.

The functionality of the video cursor control 86 provides the user with the equivalent of a mouse pick-up, put-down action. As the user moves the cursor from left to right across the screen, the user would de-activate motion-based cursor control in order to allow the user to move his head back to the left. Once the user has recentered his head, the user would once again activate the cursor control and continue to move the cursor about the screen. The activation/deactivation of the mouse input is represented by the switch 90, such that the open position of the switch disables human motion control of the cursor and supplies a zero change input to the summation operation 92 in such conditions.

Those skilled in the art should appreciate that control of scene view may also be implemented by an algorithm such as shown in Figure 3. Specifically, a similar algorithm can provide movement of the current scene view, in accord with the invention.

With video cursor control enabled, the result of the cursor update compute function 84 is added to the known current cursor position at the summing operation 92. This summation has a x component and a y component. The result of the summation 92 is used to update the cursor position (or scene view) on the user's screen via the user's operating system. Cursor position may thus be controlled by both user motion as well as the motion imparted by another input device such as a standard computer mouse.

Figure 4 provides a detailed description of the preferred implementation of the algorithm described in functions 73, 76, 78, 80 and 82 of Figure 3. Video data is received by the processing electronics in both a single frame memory 100 and a differencer 102. The output of the frame memory 100 is also provided to the

differencer 102 such that the previous frame is subtracted from the current frame. This differenced frame is than processed by a two dimensional FFT 104. The complex result of the FFT 104 is provided to a complex multiplier 106 and a complex memory 108. The complex memory 108 is the size of the processed image, each location containing both a real and imaginary component of a complex number. With each new FFT operation 104, the previous FFT result, contained in the complex memory 108, is provided to the conjugate operation 110. The complex conjugate of each element is computed and provided to the complex multiplier 106. In this manner, the FFT of the previous frame difference is conjugated and multiplied against the FFT of the current difference image.

By way of comparison between Figures 3 and 4, item 76 has similar functionality to item 102; item 78 has similar functionality to items 104, 106, 108, 110, 112; item 80 has similar functionality to item 108; and item 82 has similar functionality to item 114.

The two dimensional array of complex products output by the complex multiplier 106 is provided to a two dimensional inverse FFT operation 112. This operation creates an image of the correlation function 114 between the latest pair of difference images. The correlation image is processed by the peak detection function 114 in order to determine the shift required in aligning the two difference images. The x-y magnitude of this shift is representative of the user's motion. This x-y magnitude is provided to the software used to update the cursor position as described in Figure 3.

Figure 5 shows an algorithm process 130 of the invention and which applies motion correlation operations over sub-frames of the video image. This allows motions of various body parts to convey input with specialized meaning to applications operating on the host computer. In addition to head motion, motion of the hands, arms and legs provide for greater degrees of freedom for the user to interact with the host application (e.g., a game). Commands of this type are useful in

combative games where computer animated opponents fight under control of the user. In that instance, the hand, arm and leg motions of the user become punch, chop and kick commands to the computer after process 130. This command mode can also be used in situations where the user does not have ready access to a keyboard, to augment cursor control of the previously described head position correlator.

Process 130 identifies the functions required to derive commands from general motions of the user's body. The scene analyzer function 132 receives digitized video frames from the camera (e.g., the camera 16 of Figure 1) and identifies sub-frames within the video for tracking various parts of the user's body. The frame difference function 134 and correlator function 136 provide similar functions as processes 74, 76 and 78 of Figure 3. The correlation analyzer 138 receives correlated difference frames from the correlator function 136 and sub-frame definitions from the scene analyzer 132. The correlation analyzer 138 applies a peak detection function to each sub-frame to identify the shift required to achieve best alignment of the two images. Correlation peaks occurring in the center of the sub-frame indicate no motion, while peaks occurring elsewhere indicate the direction and magnitude of the user's motion. The motion interpreter 140 receives motion vectors for each sub-frame from the correlation analyzer 138. The motion interpreter 140 links the motion vector from each sub-frame with a particular body segment and passes this information onto the host interface 142. The host interface 142 provides for communication with the host processor (e.g., CPU 30, Figure 1). It sends data packets to the host to identify detected body motions, their directions and their amplitudes. The host interface 142 also receives instruction from the host as to which body segments to track which it then passed along to the motion interpreter 140 and the scene analyzer 132.

The scene analyzer 132 first identifies the location of the user's body in the image and locates the position of various parts of the user's body such as hands, forearms, head, and legs. The techniques and methods used to identify the user's body location and body part positions can be accomplished using techniques well

known to those skilled in the art (by way of example, via matched filtering). Body identification can also be augmented by marking different locations on the user's body with unique visual symbols. Unique symbols are assigned to key body joints such as elbows, shoulders, hands, neck, knees, and waist and are mounted on the body. See for example Figure 5A.

Figure 5A illustrates one frame 149 of data of an image of the user 150 as taken by a camera of the invention. The image corresponds to a full body image of the user 150, including arms 151, legs 152, elbows 151a, hands 153, head 154, neck 155, ears 156, and forhead 157. These parts 151-157 are identified by processes of the invention (e.g., spatial location in the image, by matched filtering or other image recognition technique), and the image is preferably marked with unique symbols (e.g., "X" for center of the face, "Y" for center of the hand 153, "T" for center of the user's foot, "Z" for body center, and "F" for forehead 157).

With further reference to Figure 5, process 130 locates various body parts and preferably marks them with symbols to fill in connecting logic (e.g., the left wrist and left elbow symbol identify the location of the left forearm). Once the user's body parts are located, sub-frames surrounding each of the body segments identified by the host processor are generated. A sub-frame is a generally regularly shaped region within the image that surrounds a particular body part. The sub-frames are sized to center the subject body part in the sub-frame and to provide enough room around the body part to accommodate typical body motions. One sub-frame 160 is shown in frame 149, Figure 5A, surrounding the user's foot "T". The scene analyzer 132 will generally not operate on each frame of video since continuously changing the sub-frames adds unnecessary complication to the correlation analyzer 138. Instead, the scene analyzer 132 runs as a background process updating the sub-frame locations periodically.

Figure 4 provides a detailed description of one algorithm which can be used to implement processes 134-138 of Figure 5.

The invention of one embodiment can thus track the motion of the user's body using symbols attached to key joints. As an example, the position of the user's left lower arm can be determined by locating the unique symbol for the left hand "Y1" and left elbow "●". Unique symbols thus allow the processor to rapidly locate each portion of the user's body in a video frame. To determine the motion of a particular part of the user's body, the algorithm (e.g., Figure 4) compares the position of the relevant body parts in consecutive frames and determines how they moved (for example, using geometry). Once motion is determined, it is then passed to the host CPU where the motion is acted on as appropriate for the particular application.

Figure 6 illustrates a two camera system 200, constructed according to the invention. The cameras 202a, 202b are arranged to view separate parts of the user: camera 202a images the user's face 204; and camera 202b images the user's hand 206. The cameras 202 conveniently rest on top of the computer display 208 coupled to the host computer 210 by cabling 216. The cameras 202 couple to the signal processing card 212 residing within the computer 210 by cabling 213. As discussed herein, motion of the user's head 204 and/or hand 206 are detected by the signal processing card 212, and difference information is communicated to the computer's CPU 210a via the computer bus 214. This difference information corresponds to composite movement of the head 204 and hand 206; and is used by the CPU 210a to command movement of display items on the display 208 (for example, the display items can include the cursor or scene view as shown on the display 208 to the user). Information shown on the display 208 is communicated from the computer 210 to the display 208 along standard cable 216.

Figures 7 and 7A illustrate how motion of a user's head is for example translated to motion of the cursor and/or scene view, in accord with the invention. Figure 7 shows a representative image 220 of a user captured within a frame of data

by a camera of the invention. Figure 7 also shows a representative image 222 (in dotted outline, for clarity of illustration) of the user in a subsequent frame of data, indicating that the user moved "M" inches. Figure 7A illustrates corresponding scene views on a computer display 224 that is coupled to processing algorithms of the invention (i.e., within a system that includes a camera that captures the images 220, 222 of Figure 7). The display 224 illustratively shows a scene view that includes a road 224a that extends off into the distance, and a house 224b adjacent to the road 224a. A computer cursor 224c is also illustratively shown on the display 224 as such a cursor is common even within computer games, providing a place for the user to select items (such as the road or house 224a, 224b) within the display 224. The display 224 also shows, with dotted outlines 226, the scene view of road and house which are shown on the display 224 after motion by the user from 220 to 222, Figure 7 (the cursor 224c is for example repositioned to position 224c'). The repositioning of the scene view from 224a, 224b to 226 occurs immediately (typically less than 1/30 second, depending upon the camera) after the movement of the user of Figure 7 from 220 to 222. The scene view is repositioned by x-pixels on the display 224, so that M/x corresponds to the magnification between user movement and scene view repositioning. This magnification can be set by parameters within the system; and can also be set by the user, if desired, at the computer keyboard. Furthermore, the rate at which the scene view moves the distance of x-pixels preferably occurs at the same rate as the rate of travel along distance M. Alternatively, the magnification can be dependent on the rate of motion such that a larger displacement of x-pixels will occur for a given motion M if the rate of change of M is larger.

Figure 8 illustrates a further motion that can be captured by a camera of the invention and processed to reposition a scene view, as shown in Figures 8A and 8B. More particularly, Figure 8 illustrates a camera 250 connected to a processing section 252 which converts user motion 254 to corresponding repositioning of the computer scene view. As above, the user 256 is captured by the camera's field of view 258 and frames of data are captured by the processing section 256. In Figure 8, motion 254 corresponds to a twisting of the user's head 256; and processing section 252 detects

this twisting and provides repositioning information to the host computer (not shown). Processing section 252 can also incorporate head-translation motion (e.g., illustrated in Figure 15A) into the scene view movement above; and can similarly reject translational movement too, if desired, so that no scene motion is observed for translation of the user 256.

Figure 8A shows a representative scene view 260 on a display 262 coupled to the host computer. Figure 8B illustrates repositioning of the scene view 260' after the processing section 252 detects motion 254 and updates the host computer with difference information (e.g., that information which the host computer uses to rotate or translate the scene view).

Figure 8A also illustrates the intent of the rotating scene view feature. In Figure 8A, a person 260a is shown in the scene view 260, except that the person 260a is almost completely obscured by the edge 262a of the display 262. By twisting the head 256 in motion 254, the scene view 262 is rotated in the corresponding direction – as shown by scene view 260' in Figure 8B – so that the user 260a' is completely visible within the scene view 260'.

Figure 8C illustrates further detail of the processing section 252. Camera data such as frames of images of a user are input to the section 252 at data port 266. The data are conditioned in the image conditioning section 268 (for example, to reduce correlated noise or other image artifacts). Thereafter, the camera data is compared and correlated in the image correlation section 270, which compares the present frame image with a series of stored images from the image memory 272. In the preferred embodiment, the present data image frame 249 is cross-correlated with each of the images within the image memory 272 to find a match. These images correspond to a series of images of the user in known positions, as illustrated in Figure 8D.

In Figure 8D, various images are stored representing various known positions of relevant part, here the user's head 256. In the position of Figure 8, for example a straight on face shot, the 0° stored memory image would provide the greatest cross-correlation value indicating a matched image position. Accordingly, the scene view would adjust to a zero position. If, however, the image correlated to a -90° position, the scene would rotate to such a position. Other movements cause additional scene view motions, including tilt and tip of the head, as shown in the two images "0°, Down 45°" image and the "0°, Up 45°". These images cause the scene view to move upwards or to tilt up or down, when the process section 252 correlates the current frame to these images. As indicated, these images have no left or right component, though other images (not shown) can certainly include left, right and tip motion simultaneously.

Figure 9 shows a system 300 constructed according to the invention and including a camera section 302 including an IR imager 304 and a camera 306, both of which view and capture frames of data from a user 308. The IR imager 304 can include, for example, a microbolometer array (i.e., "uncooled" detectors known in the art) which produces a frame of data corresponding to the infrared energy emitted from the user, such as illustrated in Figure 9A. Figure 9A shows a representative frame of IR image data 310, with zones 312 of relatively hot image data emitting from regions of forehead, nose and mouth of the user 308.

The cameras 304, 306 send image data back to the signal processing section 314. Data from the camera 306 is processed, if desired, as above, to determine difference information signal 322 used by a connected computer to reposition the cursor and/or scene view. Data from camera 304, on the other hand, is used to evaluate how much (or how hot) zones 312 appear on the user during play of the computer. The signal processing section 314 assesses the zones 312 for temperature and/or size over the course of a computer game and generate a "game speed control" signal 320 which is communicated to the user's computer (i.e., that computer used in conjunction with the system 300 of Figure 9). The user's computer

processes the signal 320 to increase or decrease the speed of a computer game in process on the computer.

Those skilled in the art should appreciate that the IR camera 304 can be used without the features of the invention which assess user movement. Rather, this aspect should be considered stand-alone, if desired, to provide active feedback into gaming speed based upon user temperature and/or stress. Note that the camera 304 can also be used to detect heartbeat since the zones 312 generally pulse at the user's heartbeat, so that heartbeat rate can also be considered as a parameter used in the generation of the game speed control signal 320. Alternatively, a pulse rate can be determined by known pulse rate systems that are physically connected to the user 308.

An IR lamp 324 can be used in system 300 to illuminate the user 308, with IR radiation 324a, such that sufficient IR illumination reflects off of the user 308 whereby motion control of the cursor and/or scene view can be made without the additional camera 306. The lamp 324 can be, and preferably is, made integrally with the section 302 to facilitate production packaging.

An IR lamp 324 operating in the near-IR can also be used with visible cameras of the invention which typically respond to near-IR wavelengths. By way of example, certain camera systems now available incorporate six IR emitters around the lens to illuminate the object without distraction to the user who cannot see the near-IR emitted light. Such a camera is suitable for use with the invention.

Figure 9B shows process methodology of the invention to process thermal user images in accord with the preferred embodiment of the invention. Specifically, a system such as system 300 first acquires a thermal image map in process block 326. This image is compared to a reference image ("REF") in process block 327. REF can either be a temperature of the user (i.e., a temperature of one hot spot of a non-stressed user, or the temperature of one hot spot of the user at an initial, pre-game

condition) or an amount of the area 312, Figure 9A, of the user in a non-stressed condition or initial pre-game condition). By way of example, REF can be an image such as the frame 310 of Figure 9A. When the temperature or area of the region 312 increases, the system 300 detects this change and determines that the image map exceeded the REF condition, as illustrated in process block 328. Should the map exceed the REF condition, the system 300 communicates this to the host processor which in turn adjusts the gaming speed, as desired. If the map does not exceed the REF condition, then the next IR image frame is acquired at block 326.

System 300 and the process steps of Figure 9B are thus suitable to adjust gaming speed in real time, depending upon user stress level. In the preferred embodiment, the gaming speed is increased automatically such that the image map exceeds the REF signal for greater than about 50% of the time, so that all users, regardless of their ability, are pushed in the particular game.

Those skilled in the art should appreciate that multi-camera embodiments of the invention can and preferably are incorporated into a common housing 338, such as shown in Figure 10. Further, as illustrated in Figure 10, cameras can also be made from detector arrays 340, processing electronics 342, and optics 344. Each camera 340, 342, 344 is constructed to process the correct electromagnetic spectrum, e.g., IR (using, for example, germanium lenses 344 and microbolometer detectors 340). Each camera has its own field of view 350a, 350b and focal distance 352a, 352b to image at least a part of the user. These field of views 350 can overlap, to view the same area such as the user's face, or they can view separate locations, such as the user's head and hand.

Cameras of the invention can also include a DSP section 356 such as described above to process user motion data. The DSP section 356 processes user motion data and sends difference information to the user's host computer. The host computer thereafter repositions the cursor and/or scene view based upon the difference information so that the user observes corresponding motion on the computer

display, as described above. Accordingly, the DSP section need not reside within the computer so long as difference information is isolated and communicated to the host computer CPU.

Those skilled in the art should appreciate that algorithms of the invention can also be processed directly by the computer's CPU, provided it has sufficient power and processing speed, to eliminate a separate DSP chip or section. DSP or equivalent processing capability can also provided within the computer's chassis by way of a computer printed circuit card installed in the chassis and connected with the camera. The location and amount of processing power, therefore, should be considered a matter of design choice and current state of the art, each technique being within the scope of the invention.

Figure 11 illustrates frame capture by one camera of the invention to isolate zones of imaging according to expected motion patterns. In Figure 11, one frame 370 of data for example covers the user's eyes 371, corresponding to one image zone; and another frame 372 of data can cover the user's head 373, corresponding to another image zone. As mentioned previously, preferably the frames 370, 372 are 64x64 pixels each, or 256x256 (or higher powers of two) to provide FFT capability on the image within the frame. A single camera can however provide both frames 370 and 372, in accord with the invention. Specifically, a dense CCD detector array (e.g., 480x740 pixels, 1000x1000 pixels, or higher) is used within the camera such that the whole array captures an image frame 376 of data, at least covering the available image format of the computer display 378. A matched filtering (or other image locate process) is processed on the frame 376 to locate the center 371a of the user's eyes (in the matched filtering process, an image data set of the user's eyes is stored in memory and correlated to the frame 376 such that a peak correlation is found at position 371a). Thereafter, a 64x64 array of data is centered about the eyes 371 to set the frame 370. To acquire the frame 372, every other pixel is discarded so that, again, a 64x64 array is set for the frame 372 (alternatively, each adjacent pair of pixels is added and averaged to provide a single number, again reducing the total number of

pixels to 64x64). Note that this process is reasonable since the width of the eyes is at least ½ the width of the user's face. Nevertheless, further compression can be obtained by utilizing every third pixel (or averaging three adjacent pixels) to obtain a larger image area in the frame 372. Note that the compression in the width and length dimensions need not be the same.

Framing of the information in Figure 11 can occur in several ways. Most cameras image at 30Hz so that image motion is smooth to the human eye. In one embodiment, one frame 370 is taken in between each frame 372, to minimize data throughput and processing; and yet to maintain dual processing of the two zones imaged in Figure 11. Alternatively, both frames 370, 372 are processed concurrently since frame 376 is typically the 30Hz frame.

Figure 11 also illustrates how framing can occur around the user's eyes 371 to acquire "blink" information to reset cursor control. A blink detected by the user's eyes in frame 370 (or other frame) can be used to (a) disable or enable control of cursor or scene movement based upon user control, or (b) simulate pick-up and replacement of the computer mouse (i.e., reinitializing movement in a particular direction). For example, by detecting a blink of the eyes 371, a system of the invention can disable human motion following control such as described herein. Another blink can be used to enable human motion following control. Blinking can also be used to continue motion in a particular direction. For example, movement of the cursor can be made to follow movement of the user's head, as described above. However, after a while, the person has to move to an uncomfortable position to keep moving the cursor (or scene). A blink can thus also serve to reposition the head back to a normal starting position so that further movement in the desired direction can be made.

Figure 12 illustrates a similar capture of a user's eyes 400, in accord with the invention. A frame 402 can thus be acquired by a camera of the invention. Figure 12A illustrates further detail of one representative frame 402, illustrating that the

user's pupils 404 are also captured. Figures 3 and 4 describe certain algorithms of the invention that are also applicable to motion of the user's pupils 404, as illustrated by left and right motion 406 and up and down motion 408. Accordingly, by zooming in on the user's eyes, another movement zone is created that causes repositioning of the cursor or the scene view based upon the movements 406, 408, much like the head movement described and illustrated in Figures 1-4.

Note that the teachings of Figures 1-4 and 12-12A can be combined within a two zone movement system so that, for example, both head motion and pupil motion can be evaluated for image motion. The cursor and/or scene view can be repositioned, therefore, based upon movements from both zones. By way of example, repositioning of items within the display (e.g., the cursor and/or scene view) can be made when the head moves but not if the head and eyes move, which might indicate that the user is simply looking elsewhere in the room due to a distraction. However, if the user moves his head, but not his eyes, he is focussed on the game and intends rotation of the scene view, in another example. Other combinations are also possible.

Cameras of the invention can also include zoom optics which (a) reduce or enlarge the image frame captured by a particular camera, or which (b) provide autofocus capability. Figure 13 shows one system 430 constructed according to the invention. A camera 432 includes camera electronics 432a and a zoom attachment 432b. Data from the camera 432 is relayed to image and interpretation feedback electronics 434 for evaluation. For purposes of image magnification, the feedback electronics serve to evaluate a given image size relative to desired image goals. For example, to image the user's eyes with high fidelity might require high density of pixels at the user's eyes (e.g., at the zone 370, Figure 11). Accordingly, the system 430 can isolate the user's eyes, such as described herein, and command the camera (through command lines 436) to increase or decrease magnification on the user's eyes so as to achieve desired resolution. The feedback electronics can also command motion of the camera to change its boresight alignment (i.e., to change where the

camera image is centered) by commanding movement of the camera when resting on one or more linear drives 438, as known in the art.

Once aligned on the desired user location, e.g., on the eyes with desired accuracy, the system 430 continues processing data such as described herein to create human interface control of items displayed on the user's computer, e.g., cursor and/or scene view. Accordingly, processing section 440 operates to detect user motion and to communicate difference information to the user's computer, as described above.

With regard to autofocus, the system 430 of Figure 13 can also be used to process user motion based upon motion towards and away from the camera. Figure 14 illustrates such a system, including a camera 450 with autofocus capability to find the best focus 452 relative to a user 454 within the field of view 456. For example, when the user 454 moves to position 460 (the user being shown in outline form 454a), the new best focus has changed to 452a. The camera 450 provides a signal 450a to the image interpretation and feedback electronics 434, Figure 13, which indicates where the user is along the "z" axis from the camera 450 to the user 454. This signal 450a is thus used much like the other motion signals described herein, to move the cursor and/or scene view in response to such movements. Figure 14A illustrates a representative scene view 462 when, for example, the user is at best focus 452. The scene view 462 includes a house image 464 with a door 465. When the user moves to position 460, the house and door 464', 465' of the scene view 462' enlarge, since the user moved closer to the camera 450. Such a motion might reveal, for example, additional objects within the house, such as illustrated by object 466, Figure 14B. Accordingly, the autofocus feature of the invention provides yet another degree of freedom in motion control, in accord with the invention.

Image data, manipulation, and human interface control can be improved, over time, by using neural net algorithms. As shown in Figure 13, a neural net update section 435 can for example couple to the feedback electronics 434 so as to

assimilate movement information and to improve data transmitted to the host computer, over time. Use of neural nets are known in the art.

Figure 15 illustrates a frame of data 490 used in accord with the invention to implement a simplified left, right, up, down movement algorithm to control cursor movement and/or scene view movement. Frame 490 is captured by a camera of the invention; and preferably the camera incorporates autofocus, as described above, to provide a crisp image of the user 492 regardless of her position within the camera's field of view. As shown, image frame 490 provides very sharp edges to the user's face, including a left edge 494a, right edge 494b, and chin 494c. These edges need only approximate vertical or horizontal position. Movement of the user results in movement of the edges 494, such as shown in Figure 15A. Figure 15A shows that once such edges are acquired, they conveniently permit subsequent movement analysis and control of scene view and/or cursor position. Specifically, Figure 15A shows movement of the user's "edges" from 494a-c to 494a-c', indicating that the user moved left (as viewed from the camera's position) and that her chin raised slightly, indicating that an upward tilt of the head. This information is assessed by the process sections as discussed above and relayed to the host computer as difference information to augment or provide cursor and/or scene movement in response to the user's movement.

Note that such edge movements roughly correspond to movement along rows and columns of the detector array. Detected movement from one row to another (or one column to another) can readily calculate the actual motion of the user from information of the user's best focus position and from the focal length of the camera's lens. This information may then be used to set the magnification of movement of items in the computer display (e.g., cursor and/or scene view).

Figure 16 illustrates an image of one object 500 used in accord with the invention to provide image manipulation in response to motion of the object. The object 500 is held by the user 501 to manipulate motion of his cursor 502 and/or

scene view 504 on his computer display 506. The object 500 is used because it exhibits an optical shape that is easily recognized through image correlation (such as matched filtering). In accord with the invention, a camera 510 is used to image the object 500; and frames of data are sent to the frame processor 512. The processor 512 determines image position - relative to a starting position - and thereafter communicates difference information to the user's computer 505 along data line 514. The difference information is used by the computer's CPU and operating system to reposition items on the display 506 in response to motion of the object 500. Almost any motion, including rotation, tilting and translation are accomplished with the object 500 relative to a start position. This start position can be triggered by the user 501 at the start of a game by commanding that the camera 510 take a reference frame ("REF") that is stored in memory 513. The user 501 commands that REF imagery be taken and stored through the keyboard 505a, connected to the computer 505, which in turn commands the processor 512 and camera 510 to take the reference frame REF.

Motion of the object 500 is thus made possible with enhanced accuracy by comparing subsequent frames of the object 500 with REF. When motion of rotation, tilt or translation are detected (for example, by using the techniques of Figures 2-4, 8-8D), then repositioning of items (502, 504) on the display 506 are follow that movement.

The techniques of the invention permit control of the scene view and/or cursor on a computer screen by motion of one or more parts of the user's body. Accordingly, as shown in Figure 17, complete motion of the user 598 can be replicated, in the invention, by correlated motion of an action figure 599 within a game. In Figure 17, user 598 is imaged by a camera 602 of the invention; and frames from the camera 602 are processed by process section 604, such as described herein. The user 598 is captured and processed, in digital imagery, and annotated with appropriate user segments, e.g., segments 1-6 indicating the user's hands, feet, head and main body. Motion of the segments 1-6 are communicated to the host computer 606 from the process section 604. The computer's operating system then updates the

associated display 608 so that the action figure 599 (corresponding to an action figure within a computer game) moves like user 598. Accordingly, user motion of action figure 599 is made by the user 598 by performing stunts (e.g., striking and kicking) that he would like the action figure 599 to perform, such as to knock out an opponent within the display 608.

In an alternative embodiment, icons can be used to simplify image and motion recognition of user segments such as segments 1-6. For example, if user 598 is marked with a star-shaped object on her hand (e.g., segment 1), then that star symbol is more easily recognized by algorithms such as described herein to determine motion. By way of another example, the hand of user 598 can be covered with a glove that has an "+" symbol on the glove. That "+" symbol can be used to more easily interpret user motion as compared to, for example, actually interpreting motion of the user's hand, which is rounded with five fingers. In a third example, user 598 can wear a article of clothing such as shirt 598a with a "+" symbol 598b; and the invention can be used to track the icon "+" 598b with great precision since it is a relatively easy object to track as compared to actual body parts. It should be apparent to those in the art that icons such as symbol 598b can be painted or pasted onto the individual too to obtain similar results.

Figure 18 illustrates a two camera system 700 used to determine translation and rotation. The forward viewing camera 702 observes the user's face 703 and determines the right-left (Δx_1) and up-down (Δy_1) translation of the user's face 703. The top viewing camera 704 observes the top of the user's face or head 705 and determines the right-left (Δx_2) and forward-backward motion (Δy_2) of the user's face or head. The two cameras 702, 704 are each processed through motion sensing algorithms 706 using the teachings above, and results are shown on the computer display 710. For purposes of illustration, the display 710 shows an image of the user; while the image can be, for example, an action figure or other computer object (including the computer cursor), as desired, which follows tracking motions Δx_1, Δy_1, Δx_2, Δy_2. As indicated in Figure 18, for example, Δy_2 can be directly applied to

motion control of the user's forward and reverse motion (note, these motions are illustrated as within a computer display 710 as processed by algorithms 706). Δx_1 can be directly applied to the users left-right sideways or strafe motion; Δy_1 can be directly applied to control the users up-down viewpoint, each as illustrated on display 710a. The results of the difference between Δx_2 and Δx_1 can be applied to control the user's left-right turn or viewpoint.

The techniques of Figure 18 can be further extended to front, side and top view cameras for complete motion detection. The top camera determines the user's left-right, front-back motion while the front facing camera determines the user's rotational up-down, left-right motion.

Figure 19 describes an algorithm to detect user eye blink. The video imagery is stored into a multiple frame buffer 800. The algorithm selects the current frame and a frame from the frame buffer and differences these frames using the adder 802. The difference frame consists of the pixel by pixel difference of the delayed frame and the current frame. The difference frame includes motion information used by the algorithms of teachings above. It also contains information on the user eye blink. The frames differenced by the adder 802 are separated temporally enough to ensure that one frame contains an image of the users face with the eyes open, the other image is of the user's face with the eyes closed. The difference image contains a two strong features, one for each eye. These features are spatially separated by the distance between the user's eyes. The blink detect function 808 inspects the image for this pair of strong features which are aligned horizontally and spaced within an expected distance based on the variation from one human face to another and the variation in seating distance expected from user to user. The recognition of the blink features may be accomplished using a matched filter or by recognition of expected frequency peaks in the frequency domain at the expected spatial frequency for human eye separation. The blink detect function 708 identifies the occurrence of a blink to a controlling function to either disable the cursor motion or take some other action.

Figure 20 illustrates a sound re-recalibration system 800 constructed according to the invention. As above, a camera 802 is arranged to view a user, a part of a user (e.g., a hand), or an object through the camera's field of view 804. A processing section 806 correlates the framing image data from camera 802 to induce movement of a scene view or cursor on the user's display 810. For purposes of illustration, the scene view or cursor is shown illustratively as a dot 808 on display 810; and movement 812 of the cursor 808 from position 808a to 808b represents a typical movement of the cursor or scene view 808 in response to movement within the field of view 804, as described above. A re-calibration section 816 is used to reset the cursor or scene view 808 back to an initial position 808a, if desired. Specifically, in one embodiment, section 816 is a microphone that responds to sound 818 generated from a sound event 818a, such as a snap of the user's fingers, or a particular word uttered by the user, to generate a signal for processing section 806 along signal line 816a; and section 806 processes the signal to move the cursor or scene view 808 back to original position 808a. In another embodiment, re-calibration section 816 can also correspond to a processing section within the processing hardware/software of system 800 to, for example, respond to the blink of a user's eyes to cause movement of the cursor 808 back to position 808a.

The invention thus attains the objects set forth above, among those apparent from the preceding description. Since certain changes may be made in the above methods and systems without departing from the scope of the invention, it is intended that all matter contained in the above description or shown in the accompanying drawing be interpreted as illustrative and not in a limiting sense. By way of example, although FFT correlation is often discussed in the above description, it should be apparent to those skilled in the art that other correlation techniques can be used with the invention to achieve similar results without departing from the scope hereof.

The following Matlab source code provides non-limiting computer code

suitable for use to control the cursor on a display such as described herein. The Matlab source code thus provides an operational demonstration of the concepts described and claimed herein. The Matlab source code is platform independent and needs only a sequence of input images. It includes a centroid operation on the correlation peak which is not included on the PC version (described below), providing a finer measurement on the motion in the image. More particularly, the centroid operation provides a refinement on locating the correlation peak. The PC code, discussed below, uses the pixel location nearest the correlation peak while the centroiding operation improves the resolution of the peak location to levels below a pixel.

```
%                   Copyright (C) 1998
%                   Video Mouse Group Partnership
%
% This following script file reads in a sequence of images of a computer user's face.
% It then processes the image sequence using the methods of difference
% frame correlation processing used for a human-computer interace.
% This code includes a centroiding operation and demonstrates the
% difference frame correlation approach.

[filename,filepath]=uigetfile('d:\videomouse\*.mat');
files=dir([filepath '*.mat']);
previousFrame=zeros(64);
inputMatrix1=zeros(64);
inputMatrix2=zeros(64);
cursorPosX(1)=0;
cursorPosY(1)=0;
for i=1:size(files,1);
  load([filepath 'frame' num2str(i)]);
        camera=conv2(double(camera'),ones(2)/4,'same');
        camera=camera(1:2:480,1:2:640);
        camera=round(camera(120-32:120+31,160-32:160+31)/4);
  if mod(i,2)
    %Compute difference frame
    inputMatrix2=camera-previousFrame;
    %Save current difference frame for next iteration
    previousFrame=camera;
    %FFT difference frame
    inputMatrix2=fft2(inputMatrix2);
   %Perform difference frame correlation by multiplying difference frame by
   %complex conjugate of previous frame
```

```
    correlationMatrix=0.00001*conj(inputMatrix2.*conj(inputMatrix1));
  else
    %Compute difference frame
    inputMatrix1=camera-previousFrame;
    %Save current difference frame for next iteration
    previousFrame=camera;
    %FFT difference frame
    inputMatrix1=fft2(inputMatrix1);
    %Perform difference frame correlation by multiplying difference frame by
    % complex conjugate of previous frame
    correlationMatrix=0.00001*conj(inputMatrix1.*conj(inputMatrix2));
  end
  %Compute inverse FFT on correlation matrix
  correlationMatrix=real(fft2(correlationMatrix));
  %Shift correlation matrix to center correlation peak
  temp=fftshift(correlationMatrix);
  %Find maximum value of correlation matrix
  correlationPeak=max(correlationMatrix(:));
  %Perform centroiding on correlation peak in order to find peak location
  %The centroiding operation is not currently incorporated into the application version
  mask=temp>.50*correlationPeak(1);
  maskSize(i)=sum(mask(:));
  if maskSize(i)<100
    colSums=sum((temp-.50*correlationPeak(1)).*mask);
    xPos(i)=sum(colSums.*(-32:31))/sum(colSums);
    rowSums=sum((temp'-.50*correlationPeak(1)).*mask');
    yPos(i)=sum(rowSums.*(-32:31))/sum(rowSums);
  else
    xPos(i)=0;
    yPos(i)=0;
  end
  if i>1
    cursorPosX(i)=cursorPosX(i-1)+xPos(i);
    cursorPosY(i)=cursorPosY(i-1)+yPos(i);
  end
end
```

The following PC source code, labeled videoMouseDlg.doc and videoMouseDSP.doc, provide non-limiting and nearly operable DSP code for control of the cursor, as described herein. The code is not smooth; and there are other files required to compile this code to an executable, as will be apparent to those skilled in the art, including header files (*.h), resource files and compiler directives.

```
*
*                    Copyright (C) 1998
*                    Video Mouse Group Partnership
*
*
*  Module  : dspcode.c

/**========================== MODIFICATION HISTORY ==========================**/
/*
*/

/**============================= MODULE ENVIRONMENT==========================**/
/*
* Include files
*/

#include   "dddefs.h"        /* XPG definitions and prototypes. */
#include   "ddptypes.h"      /* XPG function prototypes. */
#include   "dberrors.h"      /* XPG error codes. */
#include   "xpgreg.h"        /* XPG register definitions */
#include   "grabisr.h"       /* grab isr include file */
#include   "intpt40.h"       /* Parallel Runtime Support Library intertuupts header */

#include   "protocol.h"      /* Protocol constants defined for this application */

/**=============================================================================**/
/*
* Function prototypes.
*/

extern VOID    DBU_App1Interrupt (VOID);
extern VOID    DDK_PKTDelay (ULONG);

/**=============================================================================**/
/*
* Global Variables
*/
```

```
volatile LONG          G_lApp1UserMaskIntCount = 0;

/**================================ CODE ================================**/
/*=====================================================================*/
*
* Name      : DBU_Correlation
*
* Parameters :
*     lInputImage1    Image number of the 1st input image.
*     lInputImage2    Image number of the 2nd input image.
*
* Returns    :   Error status
*              P_SUCCESS
*
*  lMotionX
*  lMotionY
*  lFrameRate
*
*=====================================================================*/


/* FFT STUFF FROM TI */

#define   FFTSIZE      64              /* FFT size (n)              */
#define   LOGFFTSIZE   6               /* log(FFT size)             */
#define          FFTSIZEx2   128
#define   BLOCK0     0x002ff800       /* on-chip RAM buffer        */

extern void cfftc();                  /* C-callable complex FFT   */

float complexMatrix1[FFTSIZE][FFTSIZEx2];   /* Input matrix        */
float complexMatrix2[FFTSIZE][FFTSIZEx2];   /* Input matrix        */
float correlationMatrix[FFTSIZE][FFTSIZEx2];
long previousFrame[FFTSIZE][FFTSIZE];
float *p_localRam,*fPtr1,*fPtr2;
float correlationPeak;
float *block0    = (float *)BLOCK0, *mm1[FFTSIZE], *mm2[FFTSIZE], *mm3[FFTSIZE];



/*End FFT Stuff*/

LONG DBU_UserFunction (LONG lInputImage1, LONG lInputImage2)
{

        PROCESSINGINFO  ImageInfo1;
        PROCESSINGINFO  ImageInfo2;
```

```
        LONG    lErrorStatus;
        LONG    lFifoStatus;
        LONG    lOutputFifoStatus;
        LONG    lFrameCount = 0;
        LONG    lStatus;
        register LONG    *plAddress1;
        register LONG    *plAddress2;
        register LONG    *p_imageAddress;
        long *lPtr;
        float *currentDifference;
        float *previousDifference;

        ULONG   ulPCData;
        ULONG   ulTemp0;
        ULONG   ulValue;
        ULONG   pulPacket[5];

        long  peakRow;
        long  peakCol;
        long  row;
        long  col;
        long pixel;
        long index1;
        long index2;
        long index3;


        int targetCol= 16,targetRow = 16,targetDirection=0;

    /*---------------------------*/
    /* Initialize some variables. */
    /*---------------------------*/

        lErrorStatus = P_SUCCESS;
        ulPCData = APPLICATION_RUNNING;
        lFifoStatus = P_EMPTY;

    /*-------------------------------------------------------------*/
    /* Install the ISR found in the file INTERRUP.ASM to interrupt */
    /* resource IIOF0. Initialize the G_lApp1UserMaskIntCount.     */
    /*-------------------------------------------------------------*/

        DDF_ISRSetIIOF0 (P_INTERRUPT_USER_MASK, (VOID *)
DBU_App1Interrupt);
        G_lApp1UserMaskIntCount = 0;
```

```
        /*-------------------------------------------------------------*/
        /* Set the ID of the second image for double buffering.        */
        /* Perform a quick grab setup on the input image number.       */
        /* Call Quick Grab with no wait. This application runs the grab */
        /* in continuous mode, the grab will not return until the DMA   */
        /* has started.                                                */
        /*-------------------------------------------------------------*/

            if ((lStatus = DBF_SetSecondImageID (P_DEFAULT_QGS, lInputImage2))
             != P_SUCCESS)
            {
                    ulPCData = END_APPLICATION_REQUEST;
                    lErrorStatus = lStatus;
            } /* End if. */

            if ((lStatus = DBF_QuickGrabSetup (P_DEFAULT_QGS, lInputImage1))
            != P_SUCCESS)
            {
                     ulPCData = END_APPLICATION_REQUEST;
                    lErrorStatus = lStatus;
            } /* End if. */

            if ((lStatus = DBF_QuickGrab (P_DEFAULT_QGS, P_GRAB_INIT,
        P_GRAB_NO_WAIT))
            != P_SUCCESS)
            {
                    ulPCData = END_APPLICATION_REQUEST;
                    lErrorStatus = lStatus;
            } /* End if. */

        /*-------------------------------------*/
        /* Enable hardware interrupts on IIOF0  */
        /*-------------------------------------*/

            INT_ENABLE ();
            set_iif_flag (IIOF0_EIIOF);

        /*---------------------------------------------*/
        /* Initialize pointers to the two image buffers */
        /*---------------------------------------------*/

            if ((lStatus = DBK_MmtGetImageInfo (lInputImage1, &ImageInfo1))
            != P_SUCCESS)
            {
                     ulPCData = END_APPLICATION_REQUEST;
                    lErrorStatus = lStatus;
            } /* End if. */
```

```
        if ((lStatus = DBK_MmtGetImageInfo (lInputImage2, &ImageInfo2))
        != P_SUCCESS)
        {
                ulPCData = END_APPLICATION_REQUEST;
                lErrorStatus = lStatus;
        } /* End if. */

        plAddress1 = (LONG *) (ImageInfo1.PRO_MappedAddress);
        plAddress2 = (LONG *) (ImageInfo2.PRO_MappedAddress);

/* FFT Initialization Stuff */
        asm(" or 1800h,st");                    /* cache enable          */
/*End FFT Initialization */


  /*-------------------------------------------------*/
  /* While the input fifo from the PC does not contain  */
  /* any data, continue processing frames and returning */
  /* the results to the PC.                          */
  /*-------------------------------------------------*/


        while (ulPCData != END_APPLICATION_REQUEST){
                if (lFrameCount < (G_lApp1UserMaskIntCount)){
                        lFrameCount = G_lApp1UserMaskIntCount;

                        p_imageAddress = (G_lApp1UserMaskIntCount % 2)
                ? plAddress1 : plAddress2;
                if (G_lApp1UserMaskIntCount % 2){
                        p_imageAddress=plAddress2;*/
                        currentDifference=&complexMatrix2[0][0];
                        previousDifference=&complexMatrix1[0][0];
                }
                else{
                        p_imageAddress=plAddress1;*/
                        currentDifference=&complexMatrix1[0][0];
                        previousDifference=&complexMatrix2[0][0];
                }

/************************** Compute FFT on Difference Frame Rows ************/

                for (row=0;row<FFTSIZE;row++) {
                        for (col=0;col<FFTSIZE;col++){
                                lPtr=p_imageAddress+2*col+256*row;
                                pixel=*lPtr+*(lPtr+1)+*(lPtr+128)+*(lPtr+129);
```

```
                    /*Compute Difference Frame */
                    block0[2*col]=(float) (pixel -
previousFrame[row][col]);
                    block0[2*col+1]=0.0;
                    /*Save current frame for next iteration*/
                    previousFrame[row][col]=pixel;
                }
                cfftc(block0,FFTSIZE,LOGFFTSIZE);
                    fPtr1=currentDifference+row*FFTSIZEx2;
                for (index1=0;index1<FFTSIZEx2;index1++)
                    fPtr1[index1]=block0[index1];

            }

            for (col=0;col<FFTSIZE;col++) {

                index3=2*col;
                for (index2=0;index2<FFTSIZEx2;index2=index2+2){
                    block0[index2]=currentDifference[index3];
                    block0[index2+1]=currentDifference[index3+1];
                    index3+=FFTSIZEx2;
                }
                /*Complete column FFT of difference frame*/
                cfftc(block0,FFTSIZE,LOGFFTSIZE);


                index3=2*col;
                for(index2=0;index2<FFTSIZEx2;index2=index2+2){

                    /*Save FFT of difference frame */
                    currentDifference[index3]=block0[index2];
                    currentDifference[index3+1]=block0[index2+1];

                    block0[index2]=currentDifference[index3]
                        *previousDifference[index1]
                        +currentDifference[index3+1]
                        *previousDifference[index1+1];
                    block0[index2+1]=currentDifference[index3]
                        *previousDifference[index1+1]
                        -currentDifference[index3+1]
                        *previousDifference[index1];
                    index3+=FFTSIZEx2;
                }
                /*Compute inverse FFT for correlation matrix column*/
                cfftc(block0,FFTSIZE,LOGFFTSIZE);

                /*Save to correlation frame*/
```

```
                        fPtr1=&correlationMatrix[0][0];
                        index3=2*col;
                        for (index2=0;index2<FFTSIZEx2;index2=index2+2){
                                fPtr1[index3]=block0[index2];
                                fPtr1[index3+1]=block0[index2+1];
                                index3+=FFTSIZEx2;
                        }
                }
                correlationPeak=-100000000;
                for (row=0;row<FFTSIZE;row++) {
                        fPtr1=&correlationMatrix[row][0];
                        for (index1=0;index1<FFTSIZEx2;index1++)
                                block0[index1]=fPtr1[index1];
                /* Inverse FFT on correlation matrix row                */
                        cfftc(block0,FFTSIZE,LOGFFTSIZE);

                        fPtr1=&correlationMatrix[row][0];
                        for (col=0;col<FFTSIZE;col++){
                                index1=2*col;
                                if (correlationPeak<block0[index1]){
                                        correlationPeak=block0[index1];
                                        peakRow=row;
                                        peakCol=index1;
                                }
                                fPtr1[index1]=block0[index1];
                        }
                }

        /*-------------------------------------------------*/
        /* Send protocol, lAverage and lFrameCount to the PC. */
        /*-------------------------------------------------*/

                        pulPacket[0] = APPLICATION_RUNNING;
                        pulPacket[1] =  peakCol;
                        pulPacket[2] =  peakRow;
                pulPacket[3] = (long) (correlationPeak*.0001);
                        DDK_PKTSend (P_PACKET_USER_INTERFACE,
pulPacket,
                        4 * sizeof (LONG),P_WAITFORCOMPLETE,
                P_PC_INTERRUPT);
        } /* End if. */

        DDK_PKTInterfaceStatus (P_PACKET_USER_INTERFACE, &lFifoStatus,
                &lOutputFifoStatus);
        if (lFifoStatus != P_EMPTY){
```

```
                                DDK_PKTRecv (P_PACKET_USER_INTERFACE,
    pulPacket,
                                4 * sizeof (LONG),P_WAITFORCOMPLETE,
                                P_NO_PC_INTERRUPT);
                                ulPCData = pulPacket[0];
                        } /* End if. */
                } /* End while. */

        /*-----------------------------*/
        /* Disable the IIOF0 interrupt. */
        /*-----------------------------*/

                reset_iif_flag (IIOF0_EIIOF);
                DDF_ISRDisableIIOF0 ();

        /*----------------------------*/
        /* Abort the continuous grab. */
        /*----------------------------*/

                DBF_AbortGrab ();
                DBF_QuickGrabStatus (P_GRAB_WAIT);

        /*-------------------------------------------*/
        /* Send back a protocol word indicating the  */
        /* last packet of data. (Pad to correct size) */
        /*-------------------------------------------*/

                ulValue = APPLICATION_TERMINATED;
                DDK_PKTSend (P_PACKET_USER_INTERFACE, &ulValue, 1L,
    P_WAITFORCOMPLETE,
                        P_NO_PC_INTERRUPT);
                DDK_PKTSend (P_PACKET_USER_INTERFACE, &lErrorStatus, 1L,
                        P_WAITFORCOMPLETE, P_NO_PC_INTERRUPT);
                DDK_PKTSend (P_PACKET_USER_INTERFACE, &lErrorStatus, 1L,
                        P_WAITFORCOMPLETE,P_PC_INTERRUPT);

        /*---------------------------------------*/
        /* Empty USER input FIFO and output FIFO. */
        /* (Host won't get the END message until  */
        /* the output FIFO is empty !)           */
        /*---------------------------------------*/

                DDK_PKTInterfaceStatus (P_PACKET_USER_INTERFACE, &lFifoStatus,
                        &lOutputFifoStatus);

                while ((lFifoStatus != P_EMPTY) || (lOutputFifoStatus != P_EMPTY)){
                        if (lFifoStatus != P_EMPTY){
```

```
                                DDK_PKTRecv (P_PACKET_USER_INTERFACE,
    &ulPCData, 1L,
                                P_WAITFORCOMPLETE, P_NO_PC_INTERRUPT);
                } /* End if. */
                DDK_PKTInterfaceStatus (P_PACKET_USER_INTERFACE, &lFifoStatus,
                                &lOutputFifoStatus);
        } /* End while. */

        return P_SUCCESS;

} /* End of the DBU_UserFunction function. */

/**================================================================**/

/*
*               Copyright (C) 1998
*               Video Mouse Group Partnership
*/

/*================================================================
**
*/
// videomouseDlg.cpp : implementation file
//

#include "stdafx.h"
#include "videomouse.h"
#include "videomouseDlg.h"

#include   "dpdefs.h"       /* XPG definitions and prototypes. */
#include   "dpptypes.h"      /* XPG function prototypes. */
#include   "dberrors.h"     /* XPG error codes. */
#include   "protocol.h"     /* Protocol constants define for this application */
#include   <conio.h>        /* getch */
#include   <math.h>

static int s_runDSPLoopThreadProc;

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

#define P_ID_USER_FUNCTION1       0L
#define P_PCOUNT_USER_FUNCTION1   2
```

```
#define FRAMESIZE 64

UINT DSPLoopProc(LPVOID pclass)
{

        CVideomouseDlg& pcdd = *(reinterpret_cast<CVideomouseDlg*>(pclass));
        CString dataString;

        int savedCommandMode = DPK_XCCSetCommandType (P_USER);
        DPK_XCCSetWaitMode (P_NO_WAIT);

        DPK_XCCPushOpcode (P_ID_USER_FUNCTION1,
P_PCOUNT_USER_FUNCTION1);
        DPK_XCCPushLong ((unsigned long) pcdd.m_inputImageNumber2);      /* 2 */
        DPK_XCCPushLong ((unsigned long) pcdd.m_inputImageNumber1);      /* 1 */
        DPK_XCCSetCommandType (savedCommandMode);
        long status = DPK_XCCCheckStatus (P_PCOUT, P_XCCFIX);
        double framesPerSecond;
        time_t  start, finish;
        time( &start );
        int counter=0;
        if (status==P_SUCCESS){

                while(s_runDSPLoopThreadProc){
                        counter++;
                        status = DPK_PKTRecv (P_PACKET_USER_INTERFACE, (HVOID
*)
                                pcdd.m_DSPPacket, 4 * sizeof (LONG),
                        P_WAIT_COMPLETE);
                                long protocol = pcdd.m_DSPPacket[0];
                        dataString.Format("%d",pcdd.m_DSPPacket[1]);
                        pcdd.m_average.SetWindowText(dataString);
                        dataString.Format("%d",pcdd.m_DSPPacket[2]);
                        pcdd.m_frameNumber.SetWindowText(dataString);
                        time( &finish );
                        double elapsedTime = difftime( finish, start );
                        framesPerSecond=(double) counter/ (double)elapsedTime;
                        long max = pcdd.m_DSPPacket[3];

                        if (max>0.0){
                                double detectx=(double) pcdd.m_DSPPacket[2];
                                double detecty=0.5*(double)pcdd.m_DSPPacket[1];

                                if (detectx > 31) detectx =detectx-FRAMESIZE;
                                if (detecty > 31) detecty = FRAMESIZE-detecty;
                                else detecty = -detecty;
```

```
//                              double multiplier;/
//                              if (detectx<2) multiplier=1.0;
//                              else if (detectx<10) multiplier = exp((detectx-2.0)/2.5);
//                              else multiplier=0;

//                              detectx*=multiplier;
//                              if (detecty<2) multiplier=1.0;
//                              else if (detecty<10) multiplier = exp((detecty-2.0)/2.5);
//                              else multiplier=0;
//                              detecty*=multiplier;

                                static POINT ptCursor;
                                GetCursorPos(&ptCursor);

                                ptCursor.x-=(long) detectx;
                                ptCursor.y-=(long) detecty;

                                SetCursorPos(ptCursor.x,ptCursor.y);
                        }
                }
                savedCommandMode = DPK_XCCSetCommandType
(savedCommandMode);
                DPK_XCCSetWaitMode (P_WAIT_COMPLETE);
                DPK_EndPCK ();
                AfxMessageBox("Exited Thread");
        }
        else{
                s_runDSPLoopThreadProc=false;
                savedCommandMode = DPK_XCCSetCommandType
(savedCommandMode);
                DPK_XCCSetWaitMode (P_WAIT_COMPLETE);
                DPK_EndPCK ();
                AfxMessageBox("Exited Thread");
        }
        time( &finish );
        double elapsedTime = difftime( finish, start );
        framesPerSecond=(double) counter/ (double)elapsedTime;
        return 1;
}

////////////////////////////////////////////////////////////////////////
// CAboutDlg dialog used for App About

class CAboutDlg : public CDialog
{
public:
        CAboutDlg();
```

```
// Dialog Data
        //{{AFX_DATA(CAboutDlg)
        enum { IDD = IDD_ABOUTBOX };
        //}}AFX_DATA

        // ClassWizard generated virtual function overrides
        //{{AFX_VIRTUAL(CAboutDlg)
        protected:
        virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
        //}}AFX_VIRTUAL

// Implementation
protected:
        //{{AFX_MSG(CAboutDlg)
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
        //{{AFX_DATA_INIT(CAboutDlg)
        //}}AFX_DATA_INIT
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
        CDialog::DoDataExchange(pDX);
        //{{AFX_DATA_MAP(CAboutDlg)
        //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
        //{{AFX_MSG_MAP(CAboutDlg)
                // No message handlers
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CVideomouseDlg dialog

CVideomouseDlg::CVideomouseDlg(CWnd* pParent /*=NULL*/)
        : CDialog(CVideomouseDlg::IDD, pParent)
{
        //{{AFX_DATA_INIT(CVideomouseDlg)
        //}}AFX_DATA_INIT
        // Note that LoadIcon does not require a subsequent DestroyIcon in Win32
```

```
        m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CVideomouseDlg::DoDataExchange(CDataExchange* pDX)
{
        CDialog::DoDataExchange(pDX);
        //{{AFX_DATA_MAP(CVideomouseDlg)
        DDX_Control(pDX, IDC_FRAMENUMBER, m_frameNumber);
        DDX_Control(pDX, IDC_AVERAGE, m_average);
        //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CVideomouseDlg, CDialog)
        //{{AFX_MSG_MAP(CVideomouseDlg)
        ON_WM_SYSCOMMAND()
        ON_WM_PAINT()
        ON_WM_QUERYDRAGICON()
        ON_BN_CLICKED(IDC_ENABLE, OnEnable)
        ON_BN_CLICKED(IDC_STOP, OnStop)
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CVideomouseDlg message handlers

BOOL CVideomouseDlg::OnInitDialog()
{
        CDialog::OnInitDialog();

        // Add "About..." menu item to system menu.

        // IDM_ABOUTBOX must be in the system command range.
        ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
        ASSERT(IDM_ABOUTBOX < 0xF000);

        CMenu* pSysMenu = GetSystemMenu(FALSE);
        if (pSysMenu != NULL)
        {
                CString strAboutMenu;
                strAboutMenu.LoadString(IDS_ABOUTBOX);
                if (!strAboutMenu.IsEmpty())
                {
                        pSysMenu->AppendMenu(MF_SEPARATOR);
                        pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX,
strAboutMenu);
                }
        }
```

```
        // Set the icon for this dialog.  The framework does this automatically
        //  when the application's main window is not a dialog
        SetIcon(m_hIcon, TRUE);                 // Set big icon
        SetIcon(m_hIcon, FALSE);            // Set small icon

        InitializeFrameGrabber();

        return TRUE;  // return TRUE  unless you set the focus to a control
}

void CVideomouseDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
        if ((nID & 0xFFF0) == IDM_ABOUTBOX)
        {
                CAboutDlg dlgAbout;
                dlgAbout.DoModal();
        }
        else
        {
                CDialog::OnSysCommand(nID, lParam);
        }
}

// If you add a minimize button to your dialog, you will need the code below
//  to draw the icon.  For MFC applications using the document/view model,
//  this is automatically done for you by the framework.

void CVideomouseDlg::OnPaint()
{
        if (IsIconic())
        {
                CPaintDC dc(this); // device context for painting

                SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(), 0);

                // Center icon in client rectangle
                int cxIcon = GetSystemMetrics(SM_CXICON);
                int cyIcon = GetSystemMetrics(SM_CYICON);
                CRect rect;
                GetClientRect(&rect);
                int x = (rect.Width() - cxIcon + 1) / 2;
                int y = (rect.Height() - cyIcon + 1) / 2;

                // Draw the icon
                dc.DrawIcon(x, y, m_hIcon);
        }
```

```
        else
        {
                CDialog::OnPaint();
        }
}

// The system calls this to obtain the cursor to display while the user drags
//  the minimized window.
HCURSOR CVideomouseDlg::OnQueryDragIcon()
{
        return (HCURSOR) m_hIcon;
}

void CVideomouseDlg::OnEnable()
{
        s_runDSPLoopThreadProc = true;
        m_pDSPLoopThread = AfxBeginThread (DSPLoopProc, this);
}



void CVideomouseDlg::OnStop()
{
        if(s_runDSPLoopThreadProc){
                m_DSPPacket[0] = END_APPLICATION_REQUEST;
                DPK_PKTSend (P_PACKET_USER_INTERFACE, m_DSPPacket,4 * sizeof
                (LONG),
                        P_WAIT_COMPLETE);
                s_runDSPLoopThreadProc=false;
        }
}
#define INIT_FAILURE 1
long CVideomouseDlg::InitializeFrameGrabber()
{
                DPK_InitPCK(1);

        if ((m_status=DPK_InitXPG (0, P_IFB_RELOAD_COFF_FILE |
                        P_IFB_CHECK_REVISION,"videoMouse.out")) != P_SUCCESS){
                DPK_EndPCK ();
                m_errorMessage.Format("Error initializing FPG, status = %ld.\n", m_status);
                AfxMessageBox(m_errorMessage);
                return INIT_FAILURE;
        }
        DPK_XCCSetWaitMode (P_WAIT_COMPLETE);
        long cpsNumber = DPF_LoadCPF("vidmouse.cpf");
        if (cpsNumber != P_SUCCESS){
                m_errorMessage.Format("Error loading a CPF, status = %ld.\n", m_status);
```

```
                AfxMessageBox(m_errorMessage);
                DPK_EndPCK ();
                return INIT_FAILURE;
        }
        if ((m_status = DBF_SelectCPS (cpsNumber)) < P_SUCCESS){
                m_errorMessage.Format("Error selecting a CPS, status = %ld.\n", m_status);
                AfxMessageBox(m_errorMessage);
                DPK_EndPCK ();
                return INIT_FAILURE;
        }

        m_status=DBF_SetGrabWindow(P_DEFAULT_QGS, 256,128,176,128);

        if ((m_status = DBF_GetGrabWindow (P_DEFAULT_QGS, &m_startCol,
&m_numCols,
                &m_startRow, &m_numRows)) != P_SUCCESS){
                m_errorMessage.Format("Error DBF_GetGrabWindow: %ld.\n", m_status);
                AfxMessageBox(m_errorMessage);
                DPK_EndPCK ();
                return INIT_FAILURE;
        }

        if ((m_status = DBK_MmtCreateImage (m_numCols, m_numRows,
P_DATA_SIZE_BYTE,
                P_DATA_TYPE_INTEGER, 2, &m_inputImageNumber1,
&m_numberImagesCreated))
                != P_SUCCESS){
                m_errorMessage.Format("Error creating the input image, status = %ld.\n",
m_status);
                AfxMessageBox(m_errorMessage);
                DPK_EndPCK ();
                return INIT_FAILURE;
        }
        m_inputImageNumber2 = m_inputImageNumber1 + 1;

        return 0;
}
```

It should be understood that the following claims are to cover all generic and specific features of the invention described herein, and all statements of the scope of the invention which, as a matter of language, might be said to fall there between.

Having described the invention, what is claimed is: